Alaska
Fisheries Science
Center

National Marine
Fisheries Service

U.S DEPARTMENT OF COMMERCE

AFSC PROCESSED REPORT 2013-04

Capture-Recapture Analysis with Hidden Markov Models

December 2013

This document should be cited as follows:

Laake, J. L. 2013. Capture-recapture analysis with hidden Markov models. AFSC Processed Rep. 2013-04, 34 p. Alaska Fish. Sci. Cent., NOAA, Natl. Mar. Fish. Serv., 7600 Sand Point Way NE, Seattle WA 98115.

CAPTURE-RECAPTURE ANALYSIS WITH HIDDEN MARKOV MODELS

by

Jeffrey L. Laake

National Marine Mammal Laboratory
Alaska Fisheries Science Center
National Marine Fisheries Service, NOAA
7600 Sand Point Way NE
Seattle WA 98115
Email:jeff.laake@noaa.gov

December 2013

ABSTRACT

Hidden Markov models (HMM) provide a simple, elegant and general structure for development of existing and new models for capture-recapture analysis. These models have been used recently for relatively complex capture-recapture settings and the simplicity and generality of HMMs could be easily missed amongst the model details. To illustrate the simplicity and generality of HMMs, I describe how HMMs can be used to fit capture-recapture models starting with the simple Cormack-Jolly-Seber (CJS) model and extend to multi-state models with state uncertainty. I demonstrate the HMM algorithm for likelihood computation and explain R code that can be used to fit HMMs to make this very useful algorithm more accessible to analysts of capture-recapture data. Development of models to allow for tag loss, hierarchy of state transitions, and second-order Markov processes for breeding-feeding area transitions are relatively easily done with this framework.

CONTENTS

INTRODUCTION

Pradel (2005) describes the use of hidden Markov models (HMM) for multi-state capture-recapture models with uncertain states. His work was followed by similar applications by Ford et al. (2012), Kendall et al. (2012), and Langrock and King (2013). In each case, the models were fairly complex and the simplicity and generality of HMMs could be easily missed amongst the model details. I will attempt to elucidate both the simplicity and generality of HMMs for capture-recapture models by starting with a fairly simple example and showing how a variety of more complex models are easily derived. Hopefully this will make this very useful algorithm more accessible to analysts of capture-recapture data. I assume that the reader has a basic understanding and knowledge of capture-recapture models and statistical concepts. A table of notation is provided in the Appendix I.

An HMM is defined as an unobserved sequence $\{C_t : t = 1,2,...T\}$ that satisfies the Markov property, $\Pr(C_t \,|\, C_{t-1},..,C_1) = \Pr(C_t \,|\, C_{t-1})$ which generates an observable state-dependent sequence $\{X_t : t = 1,2,...,T\}$(Zucchini and MacDonald 2009). For many types of capture-recapture models, the observed values of the sequence (capture-history) can include the typically unknown state values (C_t). I will focus on those types of applications including Cormack-Jolly-Seber (CJS) and extensions that include multiple states and state uncertainty. Each of these models will be conditioned on the initial release in a known state but this can easily be generalized (Kendall et al. 2012).

The models and code are in the R (R Core Development Team 2012) package 'marked' (Laake et al. 2013). I will fit examples of each model using function crm. In doing so I will refer to other functions (e.g., process.data, make.design.data) in 'marked' but will not describe those functions in detail. See Laake et al. (2013) and the help in 'marked' for

further function documentation. Here I will focus on the the algorithm used to fit HMMs and the code needed to specify each of the models.

Hidden Markov Model Likelihood and Algorithm

For a detailed explanation of HMMs see Zucchini and MacDonald (2009) from which this material was drawn. Here I will provide an overview of the HMM likelihood in the specific context of capture-recapture models which are a sequence of discrete observations that typically do not assume a stationary state distribution. Let C_t be an integer from 1 to m representing the true states and x_t be an integer from 1 to s representing the observations. The integers can be attached to state labels (e.g., A or Alive for 1 and D or Dead for 2) or observation labels (e.g., 1: not seen with label 0, and 2: seen with label 1). In describing the HMM algorithm I will use the simple Cormack-Jolly-Seber (CJS) model which is used to estimate survival. For the CJS model, there are m = 2 states: Alive (1 = A) and Dead (2 = D). The s = 2 possible observations are 0 when an animal is not seen at an occasion and the state is unknown and 1 when the animal is seen at an occasion and thus known to be alive.

For CJS with a single release occasion followed by five recapture occasions, a realization of the state sequence $\{C_t : t = 1,2,...T = 6\}$ might be AAAADD. The animal was released alive on the first occasion but died in the interval between the 4th and 5th occasions. The true state sequence is assumed to be first-order Markov. For CJS, this means that at each occasion the probability the animal is alive or dead depends only on the state at the previous occasion. Typically in CJS ϕ is used to represent apparent survival probability recognizing that it includes permanent emigration. However, I will use S for survival because later ϕ is used to describe the HMM algorithm. If survival probability

(S) is constant, the four possible transition probabilities are $\Pr(C_t = \mathrm{A} \,|\, C_{t-1} = \mathrm{A}) = S$, $\Pr(C_t = \mathrm{D} \,|\, C_{t-1} = \mathrm{A}) = 1 - S$, $\Pr(C_t = \mathrm{D} \,|\, C_{t-1} = \mathrm{D}) = 1$, and $\Pr(C_t = \mathrm{A} \,|\, C_{t-1} = \mathrm{D}) = 0$. These state transition probabilities are the elements of the $m \times m$ transition probability matrix Γ. The jk^{th} element of Γ, γ_{jk} is the probability of transitioning from state j to state k. The rows of Γ must sum to 1 ($\sum_{k=1}^{m} \gamma_{jk} = 1$ for all j). For CJS, Γ is:

Γ: Transition Matrix

	Alive	Dead
Alive	S	$1-S$
Dead	0	1

.

Unless we can observe the animal at each occasion, we only partially observe the true state sequence. When the animal is seen we know it is alive, but it could be either alive or dead if it is not seen. In principle there could be 2^5 realizations of the 0-1 observation sequences but the number of possible realizations is less because for some states some observations are not possible (e.g., $\Pr(X_t = 1 | C_t = \mathrm{D}) = 0$) . For example, one possible observation sequence might be 110100. The first "1" implies that the animal was released. The animal was subsequently seen (recaptured) on the second and fourth occasions but not on the third occasion. It cannot be seen on the fifth and sixth occasions because the animal is dead ($\Pr(X_t = 1 | \mathrm{D}) = 0$) and the CJS model does not use recoveries of dead animals. These conditional probabilities of the observations given the state are used in the state-dependent probability matrix D which has s rows and m columns where the element d_{ij} is the $\Pr(X_t = i \,|\, C_t = j)$ and the columns sum to 1 ($\sum_{i=1}^{s} d_{ij} = 1$ for all j). For CJS, D is

D : State-dependent Probability Matrix

	Alive	Dead
0(not seen)	$1-p$	1
1(seen)	p	0

.

The rows of D are extracted to create an $m \times m$ diagonal matrix $P(x_t)$ where the i^{th} diagonal value is $\Pr(X_t = x_t | \mathrm{C_t} = \mathrm{i})$. For CJS, s = 2. There are two values, 0 (not seen) and 1 (seen) for x_t, so there are only two possible values of $P(x_t)$ which are

$$P(0) = \begin{bmatrix} 1-p & 0 \\ 0 & 1 \end{bmatrix}$$

and

$$P(1) = \begin{bmatrix} p & 0 \\ 0 & 0 \end{bmatrix}.$$

In addition to Γ and $P(x_t)$, the other quantity used to calculate the likelihood is δ, the row vector of m values for the initial state probability distribution, For CJS, m = 2 so δ is the row vector with elements 1 for Alive and 0 for Dead, (e.g.,$\delta = [1\ 0]$) because the animal is released alive. The likelihood for a single sequence (e.g., a single capture history) is

$$L_T = \delta P_1(x_1)\Gamma_1 P_2(x_2)\Gamma_2 P_3(x_3)...\Gamma_{T-1}P_T(x_T)1'.$$

This differs slightly from Equation 2.12 in Zucchini and MacDonald (2009) in that Γ and $P(x_t)$ have subscripts to recognize the possibility of time dependent matrices. The notation $1'$ is a column vector of 1s which when multiplied by the resulting final probability state vector, sums the probabilities of all states.

The HMM algorithm (Zucchini and MacDonald 2009) recursively computes the likelihood value by traversing along the capture history maintaining a current state row vector α_t (forward probabilities) where the j^{th} element is the joint probability $\Pr(X_1 = x_1, ...X_t =$

$x_t, C_t = j$). These forward probabilities are a sequence of vectors $\alpha_1, \alpha_2, ..., \alpha_T$ where $\alpha_1 = \delta P_1(x_1)$ and

$$\alpha_t = \delta P_1(x_1)\Gamma_1 P_2(x_2)\Gamma_2 P_3(x_3)...\Gamma_{T-1} P_T(x_T)\,.$$

It follows that $\alpha_t = \alpha_{t-1}\Gamma_{t-1}P_t(x_t)$ and $L_T = \alpha_T 1'$. To demonstrate these calculations I will show them for two simple CJS capture histories sequences with T = 3 and constant survival (S) and capture probability (p) to simplify the notation.

For CJS, the animal is released alive so $\delta = [1\ 0]$ and because the animal is released and not recaptured on the first occasion, $P_1(1)$ is by definition:

$$P_1(1) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix},$$

so $\alpha_1 = \delta P_1(1) = [1\ 0]$ for each history. First consider a capture history 100 in which the animal was released and never seen again. Because $x_2 = 0$ and $x_3 = 0$ we only need $\Gamma_t P_t(0)$ which is the same for each occasion because S and p do not change over time:

$$\Gamma P(0) = \begin{bmatrix} S & 1-S \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1-p & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} S(1-p) & 1-S \\ 0 & 1 \end{bmatrix}.$$

The value of α_2 is computed by multiplying the row vector α_1 and matrix $\Gamma P(0)$:

$$\alpha_2 = \alpha_1 \Gamma P(0) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} S(1-p) & 1-S \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} s(1-p) & 1-S \end{bmatrix}.$$

The value of α_3 is computed by multiplying α_2 and $\Gamma P(0)$:

$$\alpha_3 = \alpha_2 \Gamma P(0) = \begin{bmatrix} S(1-p) & 1-S \end{bmatrix} \begin{bmatrix} S(1-p) & 1-S \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} S^2(1-p)^2 & (1-S)S(1-p)+1-S \end{bmatrix}.$$

The sum of the elements of α_3 is the probability of observing the capture history 100 which is described in Lebreton et al. (1992) with recursive χ calculations for the tail (ending 0s)

probability.

Now consider a more interesting example capture history with a value of 101. The value of α_2 is the same as the previous capture history because each starts with 10. At occasion 2, there is a non-zero probability that the animal is dead because it was not seen. However, this is updated with the data at occasion 3. To compute, α_3 we need $\Gamma P(1)$ which is:

$$\Gamma P(1) = \begin{bmatrix} S & 1-S \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} Sp & 0 \\ 0 & 0 \end{bmatrix}$$

and α_3 is computed by multiplying α_2 and $\Gamma P(1)$:

$$\alpha_3 = \alpha_2 \Gamma P(1) = \begin{bmatrix} S(1-p) & 1-S \end{bmatrix} \begin{bmatrix} Sp & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} S^2 p(1-p) & 0 \end{bmatrix}.$$

With the addition of the last observation, the probability the animal is dead is 0 and the probability of observing the capture history 101 is $S^2 p(1-p)$. Only the forward probabilities are needed for the likelihood calculation but for predictions of the most likely state at a point in time the backward probabilities are needed. These are described in the Appendix II using these same CJS sequences.

While the above algorithm will work, it is possible for the computations to underflow numerically in computing the product of probabilities for longer sequences. Zucchini and MacDonald (2009) suggested scaling the likelihood and using an alternate algorithm to compute the logarithm of L_T. They define $\phi_t = \alpha_t/(\alpha_t 1')$ for $t = 1,...T$. Thus,

$$\phi_1 = \delta P_1(x_1)/(\delta P_1(x_1)1'),$$

and

$$\phi_t = \phi_{t-1}\Gamma_{t-1}P_t(x_t)/(\phi_{t-1}\Gamma_{t-1}P_t(x_t)1'),$$

and

$$\log L_T = \log(\delta P_1(x_1)1') + \sum_{t=2}^{T} \log\left(\phi_{t-1}\Gamma_{t-1}P_t(x_t)1'\right) .$$

In Appendix III, I describe R code that implements the above equations using the algorithm described on page 47 of Zucchini and MacDonald (2009) for their Equation 2.12 which does not assume that δ is the stationary distribution of the Markov chain.

The HMM algorithm is generic in the sense that it works for any model with an arbitrary number of states (m) and observations (s) as long as the vector δ and matrices Γ and D are defined properly. The generic nature of HMM provides a generalizable platform for developing existing and new models for capture-recapture analysis. Below I will describe the development of some currently existing models and a new model that is a special case of the model described by Kendall et al. (2012). However, to make the models useful for describing real data, I need to introduce some additional notation. Not all capture histories will have the same length. For example, with a CJS model a cohort of new animals may be released at each occasion. Thus for the first cohort, the capture history from t = 1,...T is used in the likelihood but for the remaining cohorts the capture history will be shorter. I define t* to be the first occasion used in the likelihood (e.g., t* = 2 for the second release cohort). For general models, we will want the various parameters to vary by occasion and by individual animal. Letting i index individuals and t index occasions, we can use a subscripted transition matrix Γ_{it} and state-dependent probability matrix D_{it}. In computer code, these become four dimensional (4-d) arrays which are named gamma and dmat in the R code. These have dimensions n, T, m, m and n, T, s, m respectively where n is the number of animals or number of capture histories if they are grouped. Allowing for the possibility of an estimated initial distribution that varies by animal at occasion t*, $\delta_i(x_{t*})$ becomes a matrix with dimension n, m named delta in the R code.

Developing an HMM model only requires creating code to produce the gamma, dmat and delta quantities from the specified parameters. In the 'marked' package, the models described below are "hmmCJS", "hmmMSCJS" and "hmmuMSCJS". For each of these models there is a function that creates gamma, dmat and delta from the model-specific parameters. In MARK terminology (White and Burnham 1999), parameters like survival (S) and capture probability (p) are called "real" parameters and these are computed via an inverse-link transformation from the "beta" parameters. Zucchini and MacDonald (2009) refer to these as "natural" and "working" parameters respectively. The link function is used to bound real parameters such as probabilities in the interval 0 to 1 from the beta parameters. In marked a logit link is used for probabilities such as S and p and a multinomial logit (mlogit) link is used for a set of probabilities that are constrained to sum to one. If θ_j is a real parameter and β_j is a working parameter, for a logit link $\theta_j = 1/(1+\exp(-\beta_j))$ and for an mlogit link $\theta_j = \exp(\beta_j)/\sum_k \exp(\beta_k)$ where $\sum_j \theta_j = 1$ and one of the β_j must be set to 0 to have identifiable parameters. In 'marked' for HMM models, the mlogit link is implemented with an intermediate parameter λ_j and a log-link ($\lambda_j = \exp(\beta_j)$) and setting $\lambda_j = 1$ for one of the probabilities and then normalizing by the sum (i.e., $\theta_j = \lambda_j/\sum_k \lambda_k$). Using that approach provides complete flexibility for the user in setting which real parameter is computed by subtraction (i.e., for which j $\exp(\beta_j) = 1$). The beta parameters are related to the real parameters through a design matrix which is created from a formula and a design data frame for the parameter (Laake et al. 2013). The design data frames contain a row for each animal-occasion, which enables animal and occasion-specific variation in the parameters. As an example, I will describe the design data, formula and design matrix for p with a small example with $n = 2$ and $T = 3$ and then show how this is used to populate dmat. The simplest set of design data might be

Parameter no.	Id	time
1	1	2
2	1	3
3	2	2
4	2	3

where Id specifies the animal and time is a factor variable for occasion 2 and 3 (for occasion 1 $p = 1$ and is not estimated). If the formula for p was ˜time, the design matrix would be

Parameter no.	Intercept	time
1	1	0
2	1	1
3	1	0
4	1	1

.

Let the working parameters be β_1 and β_2 for p. Then values of p are

Parameter no.	
p_1	$1/(1+exp(-\beta_1)$
p_2	$1/(1+exp(-\beta_1-\beta_2)$
p_3	$1/(1+exp(-\beta_1)$
p_4	$1/(1+exp(-\beta_1-\beta_2)$

From these parameters, dmat is a 2×2 set of 2×2 matrices as shown below:

Id	time		
1	2	$1-p_1$	1
		p_1	0
1	3	$1-p_2$	1
		p_2	0
2	2	$1-p_3$	1
		p_3	0
2	3	$1-p_4$	1
		p_4	0

.

The first two dimensions will be much larger for real data. Something similar is done for S to construct gamma. Models will have various different parameters but the general process is the same. Now I will describe some of these types of HMM models and show an example of fitting a model with 'marked'.

CORMACK-JOLLY-SEBER MODELS

The CJS model has been described already and I will not repeat its description other than to mention that in the R code survival probability (S) is named Phi to be consistent with MARK (White and Burnham 1999). The CJS model is rather simple and thus the code to compute Γ (cjs_gamma), D (cjs_dmat) and δ (cjs_delta) given in Appendix III is simple as well. Each function is passed an argument named pars which is a list structure containing a matrix for Phi and p. The values of pars are used to fill the four dimensional with id and occasion-specific transition and state-dependent probability matrices. The code sets $p = 1$ for the initial release occasion t^* (F[i] in the R code) for each animal. The cjs_delta function assigns an initial state distribution for each capture history that contains a 1 for the observed state and 0 elsewhere. For CJS the observed state is Alive. The cjs_delta function is also used for the remaining models described here because in each

case the likelihood is conditioned on the first known state of release.

The European dipper (*Cinclus cinclus*) data described by Lebreton et al. (1992) are used below as an example for fitting a model in which survival is constant and capture probability varies by time.

```
library(marked)

## This is marked 1.1.3

data(dipper)
crm(dipper,model="hmmCJS",
      model.parameters=list(Phi=list(formula=~1),
                            p=list(formula=~time)))

## Model:  HMMCJS
## Processing data

## 255 capture histories collapsed into  53

## Creating design data.
## Fitting model

##
## Elapsed time in minutes:  0.0593
##
## crm Model Summary
##
## Npar :  7
## -2lnL:  664.5
## AIC  :  678.5
##
## Beta
##                 Estimate
## Phi.(Intercept)   0.2131
## p.(Intercept)     1.2950
## p.time3           0.8013
## p.time4           0.6514
## p.time5           0.9982
## p.time6           1.4672
## p.time7           1.9955
## NULL
```

The steps that crm performed are the same as in RMark (Laake 2013) and can be done separately:

```
dp=process.data(dipper,model="hmmCJS")

## 255 capture histories collapsed into  53

# show names of list elements
names(dp)

##  [1] "data"             "model"            "mixtures"
##  [4] "freq"             "nocc"             "nocc.secondary"
##  [7] "time.intervals"   "begin.time"       "initial.ages"
## [10] "group.covariates" "start"            "ehmat"
## [13] "ObsLevels"        "fct_dmat"         "fct_gamma"
## [16] "fct_delta"        "m"

# notice functions and ObsLevels which define the model
dp$ObsLevels

## [1] 0 1

# make the design data for the parameters
ddl=make.design.data(dp)
# show design data for first 10 records
# value of p=1 when cohort=time - release occasion
head(ddl$p,10)

##    id occ time cohort age    sex freq Time Cohort Age fix
## 1   1   2    2      6   0 Female   12    0      5  -4  NA
## 2   1   3    3      6   0 Female   12    1      5  -3  NA
## 3   1   4    4      6   0 Female   12    2      5  -2  NA
## 4   1   5    5      6   0 Female   12    3      5  -1  NA
## 5   1   6    6      6   0 Female   12    4      5   0   1
## 6   1   7    7      6   1 Female   12    5      5   1  NA
## 7   2   2    2      6   0   Male   11    0      5  -4  NA
## 8   2   3    3      6   0   Male   11    1      5  -3  NA
## 9   2   4    4      6   0   Male   11    2      5  -2  NA
## 10  2   5    5      6   0   Male   11    3      5  -1  NA
```

The process.data step returns a list with the data and various attributes set for the chosen model including the code for the functions used to create dmat, gamma and delta. The make.design.data function creates a list with a design data frame for each parameter in the model. Design data can be added or a field named fix can be added or modified to fix real parameters. If the value of fix is NA the real parameter is estimated, otherwise it set

to the specified value. The processed data list and the design data can then be passed as arguments to crm as shown below:

```
crm(dp,ddl,
      model.parameters=list(Phi=list(formula=~time),
                            p=list(formula=~1)))

## Fitting model

##
## Elapsed time in minutes:  0.0357
##
## crm Model Summary
##
## Npar :  7
## -2lnL:  659.7
## AIC  :  673.7
##
## Beta
##                   Estimate
## Phi.(Intercept)  0.514372
## Phi.time2       -0.698134
## Phi.time3       -0.600897
## Phi.time4       -0.006081
## Phi.time5       -0.075709
## Phi.time6       -0.178067
## p.(Intercept)    2.220448
## NULL
```

MULTI-STATE MODELS

A multi-state model as an extension to CJS simply extends the possible states in which an animal is alive and incorporates transitions between the states. The total number of states, m, is the number of alive states plus one for dead. The initial probability vector $\delta(x_{t*})$ is 1 for the element of the first observed state and as with the CJS model the capture probability for the first occasion is fixed to 1 to condition on the first observation in the sequence. To describe the Γ and D matrices I will use $m = s = 4$ and assume state-specific

survival probability S_j $j = 1,3$ and state-specific capture probability p_j $j = 1,3$ but they can vary by occasion and individual as well.

The probability of transitioning from alive state j to alive state k (γ_{jk}), is the product of survival probability (S_j) in state j and the transition probability ψ_{jk}. If the animal does not transition to an alive state, it dies with probability $1 - S_j$ and death is an absorbing state.

Γ: Transition Matrix

	1	2	3	D
1	$S_1\psi_{11}$	$S_1\psi_{12}$	$S_1\psi_{13}$	$1-S_1$
2	$S_2\psi_{21}$	$S_2\psi_{22}$	$S_2\psi_{23}$	$1-S_2$
3	$S_3\psi_{31}$	$S_3\psi_{32}$	$S_3\psi_{33}$	$1-S_3$
D	0	0	0	1

The animal is observed to be in state j with probability p_j and is not seen with probability $1 - p_j$ and a dead animal is never "seen" because any dead recoveries are not used in this model.

D : State-dependent Probability Matrix

	1	2	3	D
0	$1-p_1$	$1-p_2$	$1-p_3$	1
1	p_1	0	0	0
2	0	p_2	0	0
3	0	0	p_3	0

This is a slightly more complicated model and the code to compute Γ and D shown below must accommodate parameters varying by stratum. The parameter list pars contains matrices for S, p and ψ. The latter uses an mlogit link to ensure $\sum_{j=1}^{m-1} \psi_{ij} = 1$ but pars only contains exp(x) (inverse log link) and these are normalized by dividing by the sum. Also, in ms_gamma two matrices are composed as shown below and then multiplied element wise together to compute Γ.

	1	2	3	D
1	S_1	S_1	S_1	$1-S_1$
2	S_2	S_2	S_2	$1-S_2$
3	S_3	S_3	S_3	$1-S_3$
D	0	0	0	1

	1	2	3	D
1	ψ_{11}	ψ_{12}	ψ_{13}	1
2	ψ_{21}	ψ_{22}	ψ_{23}	1
3	ψ_{31}	ψ_{32}	ψ_{33}	1
D	1	1	1	1

As with CJS, both Γ (gamma from ms_gamma) and D (dmat from ms_dmat) are returned as four dimensional arrays with id and occasion-specific transition and state-dependent probability matrices. The cjs_delta function described above is used to create delta. The code is in Appendix III.

As an example of a MS model, I will use the data mstrata in RMark, which has three alive states: A, B and C. I will fit the model with constant S and p and constant transition probability matrix that can vary for each of the nine possible transitions. When the design data are created for Psi, a fixed value of 1 is assigned when stratum = tostratum which fixes the transition probability for staying in the same state such that it is 1 minus the sum of the other probabilities in the mlogit link function. The value that is used can be set with subtract.stratum argument for Psi in make.design.data or can be arbitrarily selected by setting the values of fix for Psi.

```
dp=process.data(mstrata,model="hmmMSCJS")

## 252 capture histories collapsed into  252

# make the design data for the parameters
ddl=make.design.data(dp)
# show design data for first 10 records of Psi
# value of Psi=1 for staying in same state for
# identifiability with mlogit.
head(ddl$Psi[,c(1:8,12)],10)
```

```
##    id occ tostratum stratum time cohort age freq fix
## 1   1   1         A       A    1      3   0  650   1
## 2   1   1         B       A    1      3   0  650  NA
## 3   1   1         C       A    1      3   0  650  NA
## 4   1   1         A       B    1      3   0  650  NA
## 5   1   1         B       B    1      3   0  650   1
## 6   1   1         C       B    1      3   0  650  NA
## 7   1   1         A       C    1      3   0  650  NA
## 8   1   1         B       C    1      3   0  650  NA
## 9   1   1         C       C    1      3   0  650   1
## 10  1   2         A       A    2      3   0  650   1
```

```
# fit model
crm(dp,ddl,
      model.parameters=list(S=list(formula=~1),
                            p=list(formula=~1),
                            Psi=list(formula=~-1+stratum:tostratum)))
```

```
## Fitting model
```

```
##
## Elapsed time in minutes:  0.1635
##
## crm Model Summary
##
## Npar :  8
## -2lnL:  30444
## AIC  :  30460
##
## Beta
##                           Estimate
## S.(Intercept)              0.79134
## p.(Intercept)              0.02081
## Psi.stratumB:tostratumA -1.10540
## Psi.stratumC:tostratumA -1.09682
## Psi.stratumA:tostratumB -1.10524
## Psi.stratumC:tostratumB -1.09699
## Psi.stratumA:tostratumC -1.10719
## Psi.stratumB:tostratumC -1.10669
## NULL
```

MULTI-STATE MODELS WITH STATE UNCERTAINTY

If animals are seen but their state cannot always be determined with certainty, then we can introduce an observation recorded as "u" like in the implementation of the Kendall et al. (2012) model in MARK (White and Burnham 1999). For this model following Kendall et al. (2012), we need to introduce an additional parameter δ_j (different from δ used in HMM algorithm) which is the probability that the animal in state j is observed to be in state j. Thus, an animal in state j is recorded as "u" with probability $1-\delta_j$. For this model, Γ (gamma) is unchanged from the MS model and only D (dmat from function ums_dmat) changes. Following on with the example above we add a row to D (s = 5) for the "u" observation. The matrix D is similar to an MS model except now the probability that state j is recorded is $p_j\delta_j$ and the probability of being seen in state j but recorded as an unknown state is $p_j(1-\delta_j)$.

D : State-dependent Probability Matrix

	1	2	3	D
0	$1-p_1$	$1-p_2$	$1-p_3$	1
1	$p_1\delta_1$	0	0	0
2	0	$p_2\delta_2$	0	0
3	0	0	$p_3\delta_3$	0
u	$p_1(1-\delta_1)$	$p_2(1-\delta_2)$	$p_3(1-\delta_3)$	0

A simple example is observations of weaning in sea lions (*Zalophus californianus*). There are three states (m = 3): 1) S: suckling; 2) W: weaned; and 3) D: dead. Transition from suckling to weaned is considered permanent so $\psi_{WW} = 1$. There are four possible observation values (s = 4), "0", "S", "W" and "u". However, we can only observe suckling behavior ("S") and we never know for certain when the animal is weaned ($\delta_W = 0$). Thus, all observations of weaned pups will be recorded as "u". For this model, Γ and D are

Γ: Transition Matrix

	S	W	D
S	$(1-\psi_{SW})\phi_S$	$\psi_{SW}\phi_S$	$(1-\phi_S)$
W	0	ϕ_W	$(1-\phi_W)$
D	0	0	1

D : State-dependent Probability Matrix

	S	W	D
0	$1-p_S$	$1-p_W$	1
S	$p_S\delta_S$	0	0
W	0	0	0
u	$p_S(1-\delta_S)$	p_W	0

The construction of dmat is done with ums_dmat (code in Appendix III). It also uses two matrices which are constructed and then multiplied together element wise. In this case the matrices are:

	1	2	3	D
0	$1-p_1$	$1-p_2$	$1-p_3$	1
1	p_1	0	0	0
2	0	p_2	0	0
3	0	0	p_3	0
u	p_1	p_1	p_1	0

	1	2	3	D
0	1	1	1	1
1	δ_1	1	1	1
2	1	δ_2	1	1
3	1	1	δ_3	1
u	$1-\delta_1$	$1-\delta_2$	$1-\delta_3$	1

For occasion 1 and for the first occasion of each release cohort $p_j = 1$ and $\delta_j = 1$ for all j to make the likelihood conditional on first release. The initial distribution is also set with cjs_delta.

As an example, I will create some simulated data from a specified model using simHMM and then fit the same model to the simulated data. I will use the suckling-weaning example with three states ($m = 3$). In this case, the W stratum will never be observed so it is necessary to specify the strata (state) labels for the alive states. Below I define the attributes of

the simulated data by processing the data and creating and manipulating design data and specifying formula as if I was fitting the model. Once the processed data list and design data list are created a call to simHMM generates a realization from the specified model.

```
# simulate a single release cohort of 200 animals with 1 release
# and 10 recapture occasions; at least 2 unique ch are needed in simHMM
simd=data.frame(ch=c("S0000000000","SS000000000"),freq=c(100,100),stringsAsFactors=F)
# define simulation/fitting model; default for non-specified parameters is ~1
modelspec=list(p=list(formula=~stratum))
# process data with S,W strata; S=1 and W=2
sd=process.data(simd,model="hmmuMSCJS",strata.labels=c("S","W"))

## 2 capture histories collapsed into  2

# create design data
ddl=make.design.data(sd)
# set Psi transition for weaned to suckling to 0; fix is created automatically for
# Psi to fix one of the transitions to 1 for mlogit.
# default is to set Psi-xx to 1 (staying in same state)
ddl$Psi$fix[ddl$Psi$stratum==2&ddl$Psi$tostratum==1]=0
# set delta(W) to 0; non-fixed contain NA
ddl$delta$fix=ifelse(ddl$delta$stratum==2,0,NA)
# set initial parameter values for model S=~1,p=~stratum,Psi=~1,delta=~1
initial=list(S=log(.99/.01),p=c(0,-3),delta=log(.5/.5),Psi=log(0.07/.93))
# call simmHMM to get a single realization
realization=simHMM(sd,ddl,model.parameters=modelspec,initial=initial)
```

Now the same model used to generate the data is used to fit the data using the same steps but this time calling crm.

```
# take that realization and process data and fix parameters in design data
sd=process.data(realization,model="hmmuMSCJS",strata.labels=c("S","W"))

## 200 capture histories collapsed into  174

rddl=make.design.data(sd)
rddl$Psi$fix[rddl$Psi$stratum==2&rddl$Psi$tostratum==1]=0
```

```
rddl$delta$fix=ifelse(rddl$delta$stratum==2,0,NA)
# fit model
crm(sd,rddl,initial=initial,model.parameters=modelspec)

## Fitting model

##
## Elapsed time in minutes:  0.0747
##
## crm Model Summary
##
## Npar :  5
## -2lnL:  3651
## AIC  :  3661
##
## Beta
##                    Estimate
## S.(Intercept)       5.0565
## p.(Intercept)       0.1067
## p.stratumW         -3.0203
## delta.(Intercept)  -0.0454
## Psi.(Intercept)    -2.3469
## NULL
```

SUMMARY

The hidden Markov modeling approach provides a simple, elegant, and general structure for development of existing and new models for capture-recapture analysis. The code described here provides a prototype in R but most of these functions have been converted to FORTRAN in the 'marked' package to improve execution times. Conversion to Automatic Differentiation Model Builder (ADMB) (Fournier et al. 2012) would likely provide further improvements because numerical derivative computations would not be needed.

Development of models to allow tag loss, hierarchy of state transitions, and second-order Markov processes for breeding-feeding area transitions are relatively easily done with this framework.

ACKNOWLEDGMENTS

I appreciate the reviews of Tomo Eguchi and Devin Johnson which provided very useful improvements to the manuscript. A big thanks to Gary Duker and Christine Baier for their excellent editorial skills.

CITATIONS

Ford, J. H., M. V. Bravington, and J. Robbins. 2012. Incorporating individual variability into mark-recapture models. Methods Ecol. Evol. 3:1047–1054.

Fournier, D. A., H. J. Skaug, J. Ancheta, A. Magnusson, M. N. Maunder, A. Nielsen, and J. Sibert. 2012. Optimization methods and software AD model builder : using automatic differentiation for statistical inference of highly parameterized complex nonlinear models. Optim. Methods Softw. 27:233–249.

Kendall, W. L., G. C. White, J. E. Hines, C. A. Langtimm, and J. Yoshizaki. 2012. Estimating parameters of hidden Markov models based on marked individuals: use of robust design data. Ecology 93:913–920.

Laake, J. L. 2013. RMark : an R interface for analysis of capture-recapture data with MARK. AFSC Processed Rep. 2013-01, 25p. Alaska Fish. Sci. Cent., NOAA, Natl. Mar. Fish. Serv., 7600 Sand Point Way NE, Seattle WA 98115.

Laake, J. L., D. S. Johnson, and P. B. Conn. 2013. marked: An R package for maximum-likelihood and MCMC analysis of capture-recapture data. Methods Ecol. Evol. 4:885–890.

Langrock, R., King, R. 2013. Maximum likelihood estimation of mark-recapture-recovery models in the presence of continuous covariates. Ann. Appl. Stat. 7:1709-1732

Lebreton, J. D., K. P. Burnham, J. Clobert, and D. R. Anderson. 1992. Modeling survival and testing biological hypotheses using marked animals: a unified approach with case studies. Ecol. Monogr. 62:67–118.

Pradel, R. 2005. Multievent: An extension of multistate capture-recapture models to uncertain states. Biometrics 61:442–447.

R Core Development Team. 2012. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

White, G. C. and K. P. Burnham. 1999. Program MARK: survival estimation from populations of marked animals. Bird Study 46:120-139.

Zucchini, W. and I. MacDonald. 2009. Hidden Markov models for time series: An introduction using R. Chapman & Hall/CRC.

APPENDIX I

Notation

Symbol	Definition
m	number of states
s	number of observation values
t, t^*, T	t is used to index time from t = 1,...T or t=t^*,...T when sequences vary in length
C_t	state occupied by Markov chain at time t
$D(t)$	s x m matrix with ij^{th} element d_{ij} = Pr($X_t = i \,\vert\, C_t = j$)
Γ_t	m x m transition probability matrix with jk^{th} element γ_{jk} = Pr($C_{t+1} = k \,\vert\, C_t = j$)
L_T	likelihood
$P_t(x_t)$	m x m diagonal matrix; i^{th} diagonal value Pr($X_t = x_t \vert \mathrm{C_t} = \mathrm{i}$)
X_t	observation at time t
α_t	row vector of forward probabilities
β_t	row vector of backward probabilities; β also used for working parameters
δ	initial distribution of Markov chain
ϕ_t	normalized row vector of forward probabilities
p	capture probability
S	survival probability; word Phi is used in code for S in CJS

APPENDIX II

Backward Probabilities and State Prediction

Only forward probabilities are needed for the likelihood calculation but to compute predictions of the most likely state at a point in time the backward probabilities are needed. Like their counterparts, the backward probabilities are computed recursively which start with $\beta_3' = \mathbf{1}'$ and then are defined recursively as

$$\beta_{t-1}' = \Gamma \mathbf{P}(x_t)\beta_t'$$

Using a capture history of 100, we get

$$\beta_2' = \Gamma \mathbf{P}(0)\beta_3' = \begin{bmatrix} S(1-p) & 1-S \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} S(1-p)+1-S \\ 1 \end{bmatrix}$$

and

$$\beta_1' = \Gamma \mathbf{P}(0)\beta_2' = \begin{bmatrix} S(1-p) & 1-S \\ 0 & 1 \end{bmatrix} \begin{bmatrix} S(1-p)+1-S \\ 1 \end{bmatrix} = \begin{bmatrix} S^2(1-p)^2+S(1-p)(1-S)+1-S \\ 1 \end{bmatrix} .$$

Now, using the forward and backward probabilities, you can predict the most likely state (alive/dead) for each occasion. For occasion *t,* the j^{th} element of:

$$\alpha_t \beta_t / \left(\alpha_T \mathbf{1}'\right)$$

is $Pr(C_t = j | X_1 = x_1, ..., X_T = x_T)$ where vector multiplication $\alpha_t \beta_t$ is element wise. The values for a capture history of 100 are:

$$\alpha_1' \beta_1' / \left(\alpha_T \mathbf{1}'\right) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} S^2(1-p)^2+S(1-p)(1-S)+1-S \\ 1 \end{bmatrix} / S^2(1-p)^2+S(1-p)(1-S)+1-S = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\alpha_2' \beta_2' / \left(\alpha_T \mathbf{1}'\right) = \begin{bmatrix} S(1-p) \\ 1-S \end{bmatrix} \begin{bmatrix} S(1-p)+1-S \\ 1 \end{bmatrix} / S^2(1-p)^2+S(1-p)(1-S)+1-S$$

$$\alpha_3' \beta_3' / \left(\alpha_T \mathbf{1}'\right) = \begin{bmatrix} S^2(1-p)^2 \\ (1-S)S(1-p)+1-S \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} / S^2(1-p)^2+S(1-p)(1-S)+1-S$$

Expanding these out yields: probability it was alive at time 1 is:

$$Pr(C_1 = 1|X_1 = 1, X_2 = 0, X_3 = 0) = 1 \qquad ,$$

probability it was alive at time 2 is:

$$Pr(C_2 = 1|X_1 = x_1, ..., X_T = x_T) = [S^2(1-p)^2 + S(1-p)(1-S)]/[S^2(1-p)^2 + S(1-p)(1-S) + 1 - S] \qquad ,$$

and probability it was alive at time 3 is:

$$Pr(C_3 = 1|X_1 = x_1, ..., X_T = x_T) = [S^2(1-p)^2]/[S^2(1-p)^2 + S(1-p)(1-S) + 1 - S] \qquad .$$

The values for a capture history of 101 are simpler:

$$\beta_2' = \Gamma\mathbf{P}(1)\beta_3' = \begin{bmatrix} Sp & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} Sp \\ 0 \end{bmatrix}$$

$$\beta_1' = \Gamma\mathbf{P}(0)\beta_2' = \begin{bmatrix} S(1-p) & 1-S \\ 0 & 1 \end{bmatrix} \begin{bmatrix} Sp \\ 0 \end{bmatrix} = \begin{bmatrix} S^2p(1-p) \\ 0 \end{bmatrix}$$

$$\alpha_1'\beta_1'/\left(\alpha_T\mathbf{1}'\right) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} S^2p(1-p) \\ 1 \end{bmatrix} / S^2p(1-p) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\alpha_2'\beta_2'/\left(\alpha_T\mathbf{1}'\right) = \begin{bmatrix} S(1-p) \\ 1-S \end{bmatrix} \begin{bmatrix} Sp \\ 0 \end{bmatrix} / S^2p(1-p) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\alpha_3'\beta_3'/\left(\alpha_T\mathbf{1}'\right) = \begin{bmatrix} S^2p(1-p) \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} / S^2p(1-p) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

The values are all the same because the animal was seen on the last occasion and thus alive the entire time.

APPENDIX III

R code for Hidden Markov Models

The code shown here is written in R and was originally used in 'marked' but has been replaced by FORTRAN code to speed up execution time. It is shown here because most will find it more readable than the FORTRAN code. All current code for 'marked' can be found at https://github.com/jlaake/marked.

HMMLikelihood implements the algorithm described on page 47 of Zucchini and MacDonald (2009) for their Equation 2.12 which does not assume that the initial distribution is the stationary distribution of the Markov chain. HMMLikelihood computes the log-likelihood for a single sequence (x) from first which can be greater than 1 to its length T. Its arguments gamma (Γ_t for $t = 1,...,T-1$), dmat (D_t for $t = 1,...,T$) and delta (δ) are arrays computed for this sequence from the current parameter vector. The steps in the algorithm are:

1. Assign initial $\phi_{t*} = \delta P_{t*}(x_{t*})/(\delta P_{t*}(x_{t*})1')$ and lnl = $\log((\delta P_{t*}(x_{t*})1'))$
2. Loop over remaining occasions t = t*+*1,T* doing the following:

   (a) Compute v which is a product of vector ϕ_{t-1}, the matrix Γ_{t-1} (gamma[t-1,,] element *t*-1 in the gamma array) and diag(dmat[t,x[t],]) which is $P_t(x_t)$, a diagonal matrix with the diagonal elements being row x[t] of D_t from element t in the dmat array. The arrays gamma and dmat are only 3-d because they are for a specific animal.

   (b) Sum values of v into u and sum log-likelihood value log(u)

   (c) Update $\phi_t = v/u$.

```
HMMLikelihood=function(x,first,m,T,dmat,gamma,delta)
{
# Arguments:
# x: observed sequence (capture (encounter) history)
# first: occasion to start sequence
# m: number of states
# T: number of occasions; sequence length
# dmat: array of occasion specific observation probabilty matrices
# gamma: array of occasion specific transition matrices
# delta: initial state distribution
# Other variables:
# lnl: log likelihood value
# phi: alpha/sum(alpha) sequence as defined in Zucchini/MacDonald
# v: temp variable to hold phi calculations
# u: sum(v)
# Assign prob state vector for initial observation: delta*p(x_first)
  v=delta%*%diag(dmat[first,x[first],])
# Compute log-likelihood contribution for first observation; for
# models that condition on first observation u=1,lnl=0
u=sum(v)
phi=v/u
lnl=log(u)
# Loop over occasions for this encounter history (x)
for(t in (first+1):T)
{
# Compute likelihood contribution for this occasion
v=phi%*%gamma[t-1,,]%*%diag(dmat[t,x[t],])
u=sum(v)
lnl=lnl+log(u)
# Compute updated state vector
phi=v/u
}
return(lnl)
}
```

The log-likelihood for a set of capture-histories is the sum of the individual history log-likelihoods multiplied by the number of individuals with that history (freq). The R function loglikelihood does the following:

1. Computes a model-specific set of real parameters at the current parameter values (par vector) using the function reals, which multiplies the parameter vector and design matrix and applies the inverse link function for that type of parameter. It also assigns any fixed real parameters in ddl[[parname]]$fix.

2. Calls model-specific functions to compute gamma and dmat, four dimensional arrays which contain id and occasion-specific transition and state-dependent probability matrices, and delta, 2-d array containing id-specific initial state distribution.

3. Calls HMMLikelihood for each capture history (animal) (x) passing the history-specific three dimensional arrays gamma and dmat arrays and vector delta to compute the log-likelihood.

4. Returns total negative log-likelihood after multiplying by frequency of capture history.

```
loglikelihood=function(par,type,x,start,m,T,freq=1,fct_dmat,fct_gamma,
fct_delta,ddl,dml,parameters,debug=FALSE)
{
# Arguments:
# par: vector of parameter values for log-likelihood evaluation
# type: vector of parameter names used to split par vector into types
# x: matrix of observed sequences (row:id; column:occasion/time)
# start: matrix with a row for each id and two columns
#        1) first observed state, 2) first occasion observed
# m: number of states
# T: number of occasions; sequence length
# freq: vector of history frequencies or 1
# fct_dmat: function to create D from parameters
# fct_gamma: function to create gamma - transition matrix
# fct_delta: function to create initial state probability distribution matrix
# ddl: design data list of parameters for each id
# model: formulas for each parameter type
# Other variables:
# parlist: list of parameter vectors split by type (eg Phi, p in CJS)
```

```
# gamma: array of transition matrices - one for each id, time
# dmat: array of observation probability matrices - one for each id, time
#
# Create list of parameter matrices from single input parameter vector
# First split parameter vector by prameter type (type)
parlist=split(par,type)
pars=list()
# For each parameter type call function reals to compute vector
# of real parameter values; then use laply and split to create
# a matrix of parameter values with a row for each id and column for
# each occasion.
for(parname in names(parameters))
{
    R=reals(ddl=ddl[[parname]],dml=dml[[parname]],
                parameters=parameters[[parname]],parlist=parlist[[parname]])
    pars[[parname]]=laply(split(R,ddl[[parname]]$id),function(x) x)
}
# compute four dimensional arrays of id- and occasion-specific
# observation and transition matrices using parameter values
dmat=fct_dmat(pars,m,F=start[,2],T)
gamma=fct_gamma(pars,m,F=start[,2],T)
# compute matrix of initial state distribution for each id
delta=fct_delta(pars,m,F=start[,2],T,start)
# loop over each encounter history in sapply and
# create log-likelihood vector - an element for each x
# sum is total log-likelihood across individuals
# return negative log-likelihood
neglnl=-sum(freq*sapply(1:nrow(x),function(id)
            HMMLikelihood(x[id,],start[id,2],m,T,
                          dmat=dmat[id,,,],gamma=gamma[id,,,],
                          delta=delta[id,])))
return(neglnl)
}
reals=function(ddl,dml,parameters,parlist)
{
    # Computes real estimates for HMM models using inverse of
    # link from design matrix and for a particular parameter
    # type (parname); handles fixed parameters assigned by
```

```
    # non-NA value in field named fix in the ddl dataframe.
dm=dml$fe
# Currently for log,logit or identity link, return the inverse values
values=switch(parameters$link,
     log=exp(as.vector(dm%*%parlist)),
 logit=plogis(as.vector(dm%*%parlist)),
 identity=as.vector(dm%*%parlist))
    if(!is.null(ddl$time.interval))values=values^ddl$time.interval
# if some reals are fixed, set reals to their fixed values
if(!is.null(ddl$fix))
values[!is.na(ddl$fix)]=ddl$fix[!is.na(ddl$fix)]
# return vector of reals
return(values)
}
```

The above algorithm and code for computing the likelihood remains unchanged for any model, which makes generalization quite easy. Fitting a particular type of model (e.g., CJS, Multistate CJS) only requires the development of the functions fct_dmat, fct_gamma and fct_delta to create the required arrays from the parameter values. Here I will be considering models where the initial release state is known and the likelihood is conditioned on that first known state (e.g., release of a live animal in CJS). If an animal is observed in state j at occasion $t*$, then δ is a vector of 0s except the j^{th} element is 1 where j is the initial known state value x_{t*}. Also the diagonal matrix, $P_{t*}(x_{t*})$ has all zeros on the diagonal except for the element corresponding to observation of state j is 1. Thus, $\delta(x_{t*})P_1(x_{t*})1' = 1$ and its logarithm is thus 0. For these conditional models, if $t^* = T$, there is no information in the sequence.

The functions to create dmat, gamma and delta for the CJS model (hmmCJS) are cjs_dmat, cjs_gamma and cjs_delta as shown below.

```
cjs_dmat=function(pars,m,F,T)
{
# add first occasion p=1
pmat=array(NA,c(nrow(pars$p),T,2,2))
for (i in 1:nrow(pmat))
{
    pmat[i,F[i],,]=matrix(c(0,1,1,0),nrow=2,ncol=2,byrow=TRUE)
```

```
    for(j in F[i]:(T-1))
    {
        p=pars$p[i,j]
        pmat[i,j+1,,]=matrix(c(1-p,1,p,0),nrow=2,ncol=2,byrow=TRUE)
    }
}
pmat
}
cjs_gamma=function(pars,m,F,T)
{
# create four dimensional (4-d) array with a matrix for each id and occasion
# from pars$Phi which is a matrix of id by occasion survival probabilities
phimat=array(NA,c(nrow(pars$Phi),T-1,m,m))
for (i in 1:nrow(phimat))
    for(j in F[i]:(T-1))
    {
        phi=pars$Phi[i,j]
        phimat[i,j,,]=matrix(c(phi,1-phi,0,1),nrow=2,ncol=2,byrow=TRUE)
    }
phimat
}
cjs_delta=function(pars,m,F,T,start)
{
if(is.list(m))m=m$ns*m$na+1
delta=matrix(0,nrow=nrow(start),ncol=m)
delta[cbind(1:nrow(start),start[,1])]=1
delta
}
```

The functions to create dmat, gamma and delta for the Multi-state CJS model (hmmMSCJS) are ms_dmat, ms_gamma and cjs_delta. The former two functions are shown below.

```
ms_dmat=function(pars,m,F,T)
{
# create four dimensional (4-d) array with a matrix for each id and occasion from
# from pars$p which is a matrix of id by occasion x state capture probabilities
```

```
# which is split across occasions for multiple states;
# each dmat has m+1 rows (0 + m states) and m+1 columns - m states + dead
# add first occasion p=1
pmat=array(NA,c(nrow(pars$p),T,m,m))
for (i in 1:nrow(pmat))
{
    pdiag=diag(rep(1,m-1))
    pmat[i,F[i],,]=cbind(rbind(1-colSums(pdiag),pdiag),c(1,rep(0,nrow(pdiag))))
    for(j in F[i]:(T-1))
    {
        pdiag=diag(pars$p[i,((j-1)*(m-1)+1):(j*(m-1))])
        pmat[i,j+1,,]=cbind(rbind(1-colSums(pdiag),pdiag),c(1,rep(0,nrow(pdiag))))
    }
}
pmat
}
ms_gamma=function(pars,m,F,T)
{
# create an four dimensional (4-d) array with a matrix for each id and occasion for S from pars$S
# which is a matrix of id by occasion x state survival probabilities
if(is.list(m))m=m$ns*m$na+1
phimat=array(NA,c(nrow(pars$S),T-1,m,m))
for (i in 1:nrow(phimat))
{
    for(j in F[i]:(T-1))
    {
        s=pars$S[i,((j-1)*(m-1)+1):(j*(m-1))]
        smat=matrix(s,ncol=length(s),nrow=length(s))
        phimat[i,j,,]=rbind(cbind(smat,1-s),c(rep(0,length(s)),1))
    }
}
# create a 4-d array from pars$Psi which is a matrix of id by occasion x state^2
# non-normalized Psi probabilities which are normalized to sum to 1.
psimat=array(NA,c(nrow(pars$Psi),T-1,m,m))
for (i in 1:nrow(psimat))
{
    for(j in F[i]:(T-1))
    {
```

```
        psi=pars$Psi[i,((j-1)*(m-1)^2+1):(j*(m-1)^2)]
        psix=matrix(psi,ncol=sqrt(length(psi)),byrow=TRUE)
        psix=psix/rowSums(psix)
        psimat[i,j,,]=rbind(cbind(psix,rep(1,nrow(psix))),rep(1,nrow(psix)+1))
    }
}
# The 4-d arrays are multiplied and returned
phimat*psimat
}
```

The functions to create dmat, gamma and delta for the Multi-state CJS model with state uncertainty (hmmuMSCJS) are ums_dmat, ms_gamma and cjs_delta. The function ums_dmat is shown below.

```
ums_dmat=function(pars,m,F,T)
{
# create 4-d array with a p matrix for each id and occasion
# from pars$p which is a matrix of id by occasion x state capture probabilities
# which is split across occasions for multiple states;
# also and a 4-d array with delta and
# then return their product; splits across occasion for multiple states
# add first occasion p=1
pmat=array(NA,c(nrow(pars$p),T,m+1,m))
for (i in 1:nrow(pmat))
{
    pdiag=diag(rep(1,m-1))
    pmat[i,F[i],,]=cbind(rbind(1-colSums(pdiag),pdiag,colSums(pdiag)),
                         c(1,rep(0,nrow(pdiag)+1)))
    for(j in F[i]:(T-1))
    {
        pdiag=diag(pars$p[i,((j-1)*(m-1)+1):(j*(m-1))])
        pmat[i,j+1,,]=cbind(rbind(1-colSums(pdiag),pdiag,colSums(pdiag)),
                        c(1,rep(0,nrow(pdiag)+1)))
    }
}
# create 4-d array with a delta matrix for each id and occasion
# from pars$delta which is a matrix of id by occasion-state of
```

```
# probabilities of identifying state
# which is split across occasions for multiple states;
# add first occasion delta=1 for known state at release
deltamat=array(NA,c(nrow(pars$delta),T,m+1,m))
for (i in 1:nrow(deltamat))
{
    delta=rep(1,m-1)
    deltax=matrix(1,ncol=length(delta),nrow=length(delta))
    diag(deltax)=delta
    deltamat[i,F[i],,]=cbind(rbind(rep(1,ncol(deltax)),deltax,1-delta),
                             rep(1,length(delta)+2))
    for(j in F[i]:(T-1))
    {
        delta=pars$delta[i,((j-1)*(m-1)+1):(j*(m-1))]
        deltax=matrix(1,ncol=length(delta),nrow=length(delta))
        diag(deltax)=delta
        deltamat[i,j+1,,]=cbind(rbind(rep(1,ncol(deltax)),deltax,1-delta),
                                rep(1,length(delta)+2))
    }
}
# return the product of the 4-d arrays
return(pmat*deltamat)
}
```